JOURNAL OF COMPUTER AND SYSTEM SCIENCES 18, 218–227 (1979)

On *LL*-Regular Grammars

DAVID A. POPLAWSKI*

Computer Science Department, Purdue University, West Lafayette, Indiana 47907

Received February 15, 1978; revised December 13, 1979

Several results pertaining to *LL*-regular grammars are presented. The decidability of whether or not a grammar is *LL*-regular for a particular regular partition, which was first stated by Nijholt, and the undecidability of whether or not a regular partition exists for which a grammar is *LL*-regular are proved. An algorithm for converting an *LL*-regular grammar into a strong *LL*-regular grammar that generates the same language is presented, and the construction of a two pass parser is described.

1. INTRODUCTION

The syntactic analysis of programming languages has been a topic of considerable interest to compiler writers for some time. Much attention has been given to bottom up parsing methods, primarily because of the large class of grammars which can be efficiently parsed in this manner. Top down parsing methods, which are applicable to a more restricted class of grammars, have received somewhat less attention. For semantic analysis, however, the top down parsing methods are in many case prefered over bottom up methods. This preference has been demonstrated by Lewis, *et al* [6] for attributed translations and also by Knuth [5]. Consequently, extending the class of grammars parsable by top down methods can be of considerable use to the compiler writer.

Culik [2] laid the foundation for a class of top down parsing schemes by introducing *LL*(f) grammars. Intuitively, a grammar is *LL*(f) if the decision of which production to expand a given nonterminal by can be determined by applying a function f to the entire lookahead. For a practical parser, it is necessary to pick a function f which can be easily computed. A function computable by a generalized sequential machine (gsm) was chosen in [4], thereby defining the *LL*-regular class of grammars (in the definition of *LL*-regular, a regular partition is used instead of a gsm). Nijholt [7] then extended the class of *LL*-regular grammars and in addition described a practical two pass parsing algorithm.

Nijholt's parsing algorithm ([7], section 4) seems to be in error for arbitrary regular partitions, but is apparently correct for a restricted form of regular partition [8]. In this paper we define a parsing algorithm which will work for any regular partition. In addition, we prove the decidability of whether or not a grammar is *LL*-regular for a given regular partition and also prove that finding a regular partition for which a grammar is *LL*-regular is unsolvable.

* Present address: Bell Laboratories, Naperville, Illinois 60540.

0022-0000/79/030218–10$02.00/0

2. LL-Regular Grammars and Languages

We will use notation similar to the notation used in [1] plus the following: We will occasionally identify a production in P by a unique number j by writing $j\colon A \to \beta$. If $p = p_1 p_2 \cdots p_n$ is a sequence of production identifiers, then $\alpha \underset{\text{lm}}{\overset{p}{\Rightarrow}} \beta$ represents a leftmost derivation from α to β using in sequence the productions $p_1, p_2, \ldots, p_n$.

Unless otherwise indicated, the upper case letters S, A, B will be symbols in N, the upper case letter X will be a symbol in $(N \cup T)$, the lower case letters a and b will be symbols in T, the lower case letters w, x, y, z will be elements of T^*, and greek letters will be elements of $(N \cup T)^*$. If $\alpha \in (N \cup T)^*$, then α^R is the reverse of α.

Let $\pi = \{R_1, \ldots, R_n\}$ be a collection of non-empty regular sets over T. If the R_i are pairwise disjoint and the union of all R_i in π is equal to T^*, then π is called a regular partition of T^*. For strings $x, y \in T^*$, if $x \in R_i$ and $y \in R_i$ for some i, then we will write $x \equiv y \pmod{\pi}$.

We first define two classes of grammars. Both classes generate the LL-regular languages. The first is equivalent to the definition of LL-regular grammars given in [4]. The second was given in [7] and is more general class of grammars. Since they are analogous to the definitions of strong $LL(k)$ and $LL(k)$ grammars, these two grammar classes are called strong LL-regular and LL-regular respectively.

Definition 2.1. Let $G = (N, T, P, S)$ be a cfg. Let $\pi = \{R_1, \ldots, R_n\}$ be a regular partition of T^*. G is said to be strong LL-regular for π if, given any two leftmost derivations:

$$S \underset{\text{lm}}{\overset{*}{\Rightarrow}} w_1 A \alpha_1 \underset{\text{lm}}{\Rightarrow} w_1 \beta \alpha_1 \underset{\text{lm}}{\overset{*}{\Rightarrow}} w_1 x,$$

$$S \underset{\text{lm}}{\overset{*}{\Rightarrow}} w_2 A \alpha_2 \underset{\text{lm}}{\Rightarrow} w_2 \gamma \alpha_2 \underset{\text{lm}}{\overset{*}{\Rightarrow}} w_2 y$$

such that $x \equiv y \pmod{\pi}$, then it follows that $\beta = \gamma$.

Definition 2.2. Let $G = (N, T, P, S)$ be a cfg. Let $\pi = \{R_1, \ldots, R_n\}$ be a regular partition of T^*. G is said to be LL-regular for π (written $LL(\pi)$) if, given any two leftmost derivations:

$$S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\beta\alpha \underset{\text{lm}}{\overset{*}{\Rightarrow}} wx,$$

$$S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\gamma\alpha \underset{\text{lm}}{\overset{*}{\Rightarrow}} wy$$

such that $x \equiv y \pmod{\pi}$, then it follows that $\beta = \gamma$.

The same class of languages is generated by strong LL-regular and LL-regular grammars (see section 4). This class is defined as follows:

Definition 2.3. A language L over alphabet T is said to be LL-regular if there exists a grammar G and a regular partition π of T^* such that G is $LL(\pi)$ and $L = L(G)$.

3. Decidability

In this section we present two decidability results for LL-regular grammars. First we show that it is decidable whether or not a grammar is $LL(\pi)$ for a specific regular partition π. This result was stated but not proven by Nijholt [7]. Second, we show that it is not decidable whether or not there exists a regular partition for which a grammar is LL-regular. The proof of this theorem uses a separability argument from [3] that is similar to one used in the analogous proof for strong LL-regular grammars [4].

The next proposition establishes a procedure for deciding whether a grammar is LL-regular or not. Without loss of generality, we assume that the grammar is reduced.

Lemma 3.1. *Let $G = (N, T, P, S)$ be a cfg. Let $\pi = \{R_1, ..., R_n\}$ be a regular partition of T^*. Let*

$$L_{ij} = \{\alpha \in (N \cup T)^* \mid S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\beta\alpha \underset{\text{lm}}{\overset{*}{\Rightarrow}} wx,\ w \in T^*,\ x \in R_i,\ j\colon A \to \beta \text{ in } P\}.$$

G is $LL(\pi)$ if and only if $L_{ij} \cap L_{ik} = \phi$ for all R_i in π and productions $j\colon A \to \beta$ and $k\colon A \to \gamma$ in P such that $\beta \neq \gamma$.

Proof. (if) Consider the derivations

$$S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\beta\alpha \underset{\text{lm}}{\overset{*}{\Rightarrow}} wx, \tag{1}$$

$$S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\gamma\alpha \underset{\text{lm}}{\overset{*}{\Rightarrow}} wy, \tag{2}$$

where $j\colon A \to \beta$, $k\colon A \to \gamma$ and $x \equiv y \pmod{\pi}$ (assume $x, y \in R_i$). Derivation (1) implies that $\alpha \in L_{ij}$, while derivation (2) implies that $\alpha \in L_{ik}$. Since $L_{ij} \cap L_{ik} = \phi$ when $\beta \neq \gamma$, we must have $\beta = \gamma$. Therefore G is $LL(\pi)$.

(only if) Assume $L_{ij} \cap L_{ik} \neq \phi$ $(j \neq k)$. Let α be any element of $L_{ij} \cap L_{ik}$. Since $\alpha \in L_{ij}$, there must be a derivation using $j\colon A \to \beta$

$$S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\beta\alpha \underset{\text{lm}}{\overset{*}{\Rightarrow}} wx,$$

where $x \in R_i$. The similar derivation

$$S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\gamma\alpha \underset{\text{lm}}{\overset{*}{\Rightarrow}} wy$$

using $k\colon A \to \gamma$ must also exist, with $\alpha \in L_{ik}$ assuring that $y \in R_i$. Consequently $x \equiv y \pmod{\pi}$. Since G is $LL(\pi)$, $\beta = \gamma$ and $j = k$, which is a contradiction.

In order to satisfy the conditions of Lemma 3.1 we must construct the sets L_{ij}. To that end, consider the following construction (where $G = (N, T, P, S)$ and $\pi = \{R_1, ..., R_n\}$ is a regular partition of T^*):

For each production $j\colon A \to \beta$ in P define the following cfg:

$$G_j = (N_j, T, P_j, S_j),$$

where $N_j = N \cup \{X' \mid X \in N\} \cup \{S_j\}$,

$$P_j = P \cup \{S_j \to S'\} \cup \{A' \to \beta \mid j\colon A \to \beta \in P\}$$
$$\cup \{X' \to X_i' X_{i+1} \cdots X_s \mid X \to X_1 X_2 \cdots X_s \in P \text{ for all } 1 \leqslant i \leqslant s,\ X_i \in N\}.$$

PROPOSITION 3.2. $L(G_j) = \{x \in T^* \mid S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\beta\alpha \underset{\text{lm}}{\overset{*}{\Rightarrow}} wx$ *is a derivation in* G *using production* $j\colon A \to \beta\}$.

Let $M_i = (Q, T, \delta, q_0, F)$ be the minimal deterministic finite automaton accepting the language R_i for $1 \leqslant i \leqslant n$. For each cfg G_j from above and for each R_i in π define the following cfg :

$$G_{ij} = (N_{ij}, T, P_{ij}, S_{ij}),$$

where

$$N_{ij} = \{S_{ij}\} \cup \{[p, X, q] \mid X \in N_j \text{ and } p, q \in Q\},$$
$$P_{ij} = \{S_{ij} \to [q_0, S_j, p] \mid p \in F\}$$
$$\cup \{[p, X, q] \to [p, X_1, q_1][q_1, X_2, q_2] \cdots [q_{n-1}, X_n, q]$$
$$X \to X_1 X_2 \cdots X_n \in P_j \text{ and for all } q_1, \ldots, q_{n-1} \in Q\}$$
$$\cup \{[p, A, p] \to \epsilon \mid (A \to \epsilon) \in P_j\}$$
$$\cup \{[p, a, q] \to a \mid a \in T \text{ and } \delta(p, a) = q\}.$$

The grammars G_{ij} are then put into reduced form by eliminating useless productions and non-terminals.

PROPOSITION 3.3. $L(G_{ij}) = \{x \in R_i \mid S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\beta\alpha \underset{\text{lm}}{\overset{*}{\Rightarrow}} wx$ *is a derivation in* G *using production* $j\colon A \to \beta\}$.

Define the homomorphisms h_{ij} from N_{ij} to N_j where $h_{ij}([p, X, q]) = X$. For each cfg G_{ij} above define the following left-linear grammar:

$$H_{ij} = (N_{ij}, N \cup T, P'_{ij}, S_{ij}),$$

where

$$P'_{ij} = \{[p_1, X, p_{n+1}] \to [p_1, X_1, p_2]X_2 \cdots X_n \mid$$
$$[p_1, X, p_{n+1}] \to [p_1, X_1, p_2] \cdots [p_n, X_n, p_{n+1}] \in P_{ij}\}$$
$$\cup \{[p, A', q] \to \epsilon \mid [p, A', q] \to \beta \in P_{ij} \text{ and } A' \to h_{ij}(\beta)$$
$$\text{is the production in } P_j \text{ obtained from } j\colon A \to \beta \in P\}.$$

PROPOSITION 3.4. $L(H_{ij}) = \{\alpha \mid S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\beta\alpha \underset{\text{lm}}{\overset{*}{\Rightarrow}} wx$ *is a derivation in* G *using production* $j\colon A \to \beta$ *and* $x \in R_i\} = L_{ij}$.

The decidability result now follows from proposition 3.1 and the constructions above.

THEOREM 3.5. *Let $G = (N, T, P, S)$ be a cfg and let $\pi = \{R_1, \ldots, R_n\}$ be a regular partition of T^*. It is decidable whether G is $LL(\pi)$ or not.*

Proof. The construction of the left-linear grammars H_{ij} above define the regular sets $L_{ij} = L(H_{ij})$. Since there are a finite number of regular sets L_{ij}, there are a finite number of intersections $L_{ij} \cap L_{ik}$. The L_{ij} are regular so it is decidable whether each intersection is empty or not. Therefore the decision procedure for $LL(\pi)$-ness consists of testing $L_{ij} \cap L_{ik}$ for emptiness for all i, j, k such that $j: A \to \beta$, $k: A \to \gamma$ and $\beta \neq \gamma$. By proposition 3.1, if any intersection is non-empty, G is not $LL(\pi)$; if all intersections are empty, G is $LL(\pi)$.

Although it is decidable whether a grammar is LL-regular for a given regular partition π, it is not decidable whether a grammar has a regular partition for which it is LL-regular. The following lemma will help establish this fact.

LEMMA 3.6. *If $G = (N, T, P, S)$ is LL-regular for the regular partition $\pi = \{R_1, \ldots, R_n\}$ then for any two productions $A \to \beta$ and $A \to \gamma$ in P $(\beta \neq \gamma)$, there exists a regular set R such that $\{x \mid S \overset{*}{\underset{\text{lm}}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\beta\alpha \overset{*}{\underset{\text{lm}}{\Rightarrow}} wx\} \subseteq R$ and $\{y \mid S \overset{*}{\underset{\text{lm}}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\gamma\alpha \overset{*}{\underset{\text{lm}}{\Rightarrow}} wy\} \cap R = \phi$.*

Proof. R is the union of all sets R_i in π such that $R_i \cap \{x \mid S \overset{*}{\underset{\text{lm}}{\Rightarrow}} wA\alpha \underset{\text{lm}}{\Rightarrow} w\beta\alpha \overset{*}{\underset{\text{lm}}{\Rightarrow}} wx\} \neq \phi$.

THEOREM 3.7. *It is not decidable whether or not a context-free grammar $G = (N, T, P, S)$ is LL-regular.*

Proof. Let $G_1 = (N_1, T_1, P_1, S_1)$ and $G_2 = (N_2, T_2, P_2, S_2)$ be any two cfg's where $N_1 \cap N_2 = \phi$. Let $G = (N_1 \cup N_2 \cup \{S\}, T_1 \cup T_2, P_1 \cup P_2 \cup \{S \to S_1, S \to S_2\}, S\}$. By the previous lemma, if G has a regular partition π such that G is $LL(\pi)$ then there exists a regular set R such that for productions $S \to S_1$ and $S \to S_2$, $\{x \mid S \overset{*}{\underset{\text{lm}}{\Rightarrow}} S \underset{\text{lm}}{\Rightarrow} S_1 \overset{*}{\underset{\text{lm}}{\Rightarrow}} x\} = L(G_1) = L_1 \subseteq R$ and $\{y \mid S \overset{*}{\underset{\text{lm}}{\Rightarrow}} S \underset{\text{lm}}{\Rightarrow} S_2 \overset{*}{\underset{\text{lm}}{\Rightarrow}} y\} = L(G_2) = L_2 \cap R = \phi$. Therefore, being able to decide whether or not there exists a π for which G is $LL(\pi)$ implies being able to decide whether or not there is a regular separating set for the cfg's L_1 and L_2. This problem was reported in [3] and [9] to be undecidable. Consequently, it is undecidable whether or not π exists.

4. CONVERTING LL-REGULAR GRAMMARS TO STRONG LL-REGULAR GRAMMARS

The class of LL-regular grammars properly includes the strong LL-regular grammars [7], but the class of LL-regular and strong LL-regular languages are the same. This is because for any LL-regular grammar one can construct a strong LL-regular grammar that generates the same language. In this section we describe an algorithm for constructing a strong LL-regular grammar generating the same language as a given LL-regular grammar.

The strong LL-regular grammar $G' = (N', T, P', S')$ is constructed from the LL-regular grammar $G = (N, T, P, S)$ such that G' left-covers [1] G.

CONSTRUCTION 4.1. Let $G = (N, T, P, S)$ be a cfg and let $\pi = \{R_1, \ldots, R_n\}$ be a regular partition of T^*. Let m be the number of productions in P. Let $M_{ij} = (Q_{ij}, T, \delta_{ij}, q^0_{ij}, F_{ij})$ be the minimal deterministic finite automaton accepting L^R_{ij} for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$. Let $Q = \{[q_{1,1}, q_{1,2}, \ldots, q_{n,m}] \mid q_{i,j} \in Q_{ij}\}$ and let $q^0 = [q^0_{1,1}, q^0_{1,2}, \ldots, q^0_{n,m}]$. Define the function $\delta([q_{1,1}, \ldots, q_{n,m}], X) = [\delta_{1,1}(q_{1,1}, X), \ldots, \delta_{n,m}(q_{n,m}, X)]$ for $X \in N \cup T$ and the projections $h_{ij}([q_{1,1}, \ldots, q_{n,m}]) = q_{i,j}$ for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$. Then $G' = (N', T, P', S')$, where $N' = \{[X, q] \mid X \in (N \cup T), q \in Q\}$, $S' = [S, q^0]$ and $P' = \{[X, q] \to [X_1, q_1][X_2, q_2] \cdots [X_s, q_s] \mid X \to X_1 X_2 \cdots X_s \in P, q_1, q_2, \ldots, q_s \in Q, q_s = q, q_{i-1} = \delta(q_i, X_i)\} \cup \{[A, q] \to \epsilon \mid A \to \epsilon \in P\} \cup \{[a, q] \to a \mid a \in T, q \in Q\}$.

PROPOSITION 4.1. *The grammar G' constructed above left-covers G.*

Proof. Straightforward induction on the lengths of derivations results in showing that $L(G) = L(G')$. The homomorphism h defined as $h([X, q] \to [X_1, q_1] \cdots [X_s, q_s]) = X \to X_1 \cdots X_s$ and $h([a, q] \to a) = \epsilon$ satisfies the remaining two conditions for a left-cover.

PROPOSITION 4.2. *If the grammar G is LL-regular for π, then the grammar G' constructed above is strong LL-regular for π.*

Proof. For the purposes of this proof, the strings α, α_1, α_2, β and γ are obtained from the strings α', α'_1, α'_2, β' and γ' by deleting the state component from each symbol in the string.

We first notice that by the construction of G' if $[X, q] \to [X_1, q_1] \cdots [X_s, q_s]$ then $q_1 = \delta(q_s, X_s \cdots X_2)$ and straightforward induction on the length of a derivation shows that if $[S, q_0] \overset{*}{\underset{\text{lm}}{\Rightarrow}} w[A, q]\alpha'$ then $q = \delta(q_0, \alpha^R)$. We also notice that if $[A, q]\alpha' \overset{}{\underset{\text{lm}}{\Rightarrow}} \beta'\alpha' \overset{*}{\underset{\text{lm}}{\Rightarrow}} x$ and $x \in R_i$ then $h_{ij}(q) \in F_{ij}$ which implies that $\alpha \in L(H_{ij})$.

Now consider the derivations

$$[S, q_0] \overset{*}{\underset{\text{lm}}{\Rightarrow}} w_1[A, q]\alpha'_1 \underset{\text{lm}}{\Rightarrow} w_1\beta'\alpha'_1 \overset{*}{\underset{\text{lm}}{\Rightarrow}} w_1 x,$$

$$[S, q_0] \overset{*}{\underset{\text{lm}}{\Rightarrow}} w_2[A, q]\alpha'_2 \underset{\text{lm}}{\Rightarrow} w_2\gamma'\alpha'_2 \overset{*}{\underset{\text{lm}}{\Rightarrow}} w_2 y$$

in G' and let $x, y \in R_i$ (that is $x \equiv y \pmod{\pi}$). We want to show that $\beta' = \gamma'$. Consider the corresponding derivations in G:

$$S \overset{*}{\underset{\text{lm}}{\Rightarrow}} w_1 A\alpha_1 \underset{\text{lm}}{\Rightarrow} w_1\beta\alpha_1 \overset{*}{\underset{\text{lm}}{\Rightarrow}} w_1 x,$$

$$S \overset{*}{\underset{\text{lm}}{\Rightarrow}} w_2 A\alpha_2 \underset{\text{lm}}{\Rightarrow} w_2\gamma\alpha_2 \overset{*}{\underset{\text{lm}}{\Rightarrow}} w_2 y,$$

where j: $A \to \beta$, k: $A \to \gamma$ and $x, y \in R_i$. We know that $\alpha_1 \in L(H_{ij})$, $\alpha_2 \in L(H_{ik})$ and since G is $LL(\pi)$, $L(H_{ij}) \cap L(H_{ik}) = \phi$ unless $j = k$. But the derivations in G' insure that state q is identical in both $[A, q]$'s. This can occur only if $j = k$. Therefore $\beta = \gamma$.

The construction then insures that $\beta' = \gamma'$ since the state components are uniquely determined once q has been fixed. Consequently G' is strong LL-regular for π.

5. An LL-Regular Parsing Algorithm

Nijholt [7] shows how to construct a parser for LL-regular grammars. The construction seems to depend on the regular partition being a congruence [8] (that is, if $x \equiv y \pmod{\pi}$ then $axb \equiv ayb \pmod{\pi}$ for any $a, b \in T \cup \{\epsilon\}$). The following example shows that if the partition is not a congruence then the statement of Lemma 1a of [7] does not hold. Consequently the resulting parser may be incorrect. The following definitions apply:

$$\text{BLOCK}(\alpha) = \{R_k \in \pi \mid L(\alpha) \cap R_k \neq \phi\}.$$
$$R_i \square R_j = \{R_k \in \pi \mid R_k \cap (R_i \cdot R_j) \neq \phi\}.$$
$$\text{If } L_1, L_2 \subseteq \pi \text{ then } L_1 \square L_2 = \{R_k \in \pi \mid R_k \in R_i \square R_j,\ R_i \in L_1,\ R_j \in L_2\}.$$

Lemma 1a of [7] states that $\text{BLOCK}(\alpha\beta) = \text{BLOCK}(\alpha) \square \text{BLOCK}(\beta)$, where $\alpha, \beta \in (N \cup T)^*$. Consider the following grammar and regular partition.

Let $G = (\{S, X, Y, Z, F\}, \{a, b, c\}, P, S)$ where $P = \{S \to ZF, Z \to X, Z \to Y, X \to aa, Y \to ac, F \to ba\}$. Let $\pi = \{R_0, R_1, R_2, R_3, R_4, R_5, R_6\}$ where $R_0 = \{\epsilon\}$, $R_1 = \{aa\}$, $R_2 = \{ba, bc\}$, $R_3 = \{aaba\}$, $R_4 = \{aabc, acba, acbc\}$, $R_5 = \{ac\}$, $R_6 = T^* - (R_0 \cup R_1 \cup R_2 \cup R_3 \cup R_4 \cup R_5)$. Notice that $ba \equiv bc \pmod{\pi}$ but $aaba \not\equiv aabc \pmod{\pi}$, so π is not a congruence.

Consider now the proposition that $\text{BLOCK}(XF) = \text{BLOCK}(X) \square \text{BLOCK}(F)$.

$$\begin{aligned}\text{BLOCK}(XF) &= \{R_k \in \pi \mid R_k \cap L(XF) \neq \phi\} \\ &= \{R_k \in \pi \mid R_k \cap \{aaba\} \neq \phi\} \\ &= \{R_3\}.\end{aligned}$$

$$\begin{aligned}\text{BLOCK}(X) \square \text{BLOCK}(F) &= \{R_k \in \pi \mid R_k \cap L(X) \neq \phi\} \square \{R_k \in \pi \mid R_k \cap L(F) \neq \phi\} \\ &= \{R_k \in \pi \mid R_k \cap \{aa\} \neq \phi\} \square \{R_k \in \pi \mid R_k \cap \{ba\} \neq \phi\} \\ &= \{R_1\} \square \{R_2\} \\ &= \{R_k \in \pi \mid R_k \in R_1 \square R_2\} \\ &= \{R_k \in \pi \mid R_k \in \{R_j \in \pi \mid R_j \cap (R_1 \cdot R_2) \neq \phi\}\} \\ &= \{R_k \in \pi \mid R_k \in \{R_j \in \pi \mid R_j \cap \{aaba, aabc\} \neq \phi\}\} \\ &= \{R_k \in \pi \mid R_k \in \{R_3, R_4\}\} \\ &= \{R_3, R_4\}.\end{aligned}$$

Therefore we have that $\text{BLOCK}(XF) = \{R_3\} \neq \{R_3, R_4\} = \text{BLOCK}(X) \square \text{BLOCK}(F)$. As a result, the use of $\text{BLOCK}(\alpha)$ in the construction of the parser may cause an undefined action when in fact none should occur.

We now present a method for constructing a parser for LL-regular grammars which does not require that the regular partition be a congruence. The parser is similar to the one in [7], but is constructed in a way that avoids the problem illustrated above.

An *LL*-regular grammar will be parsed using a two pass algorithm. The first pass does a transformation of the input by processing it in reverse order. The transformed input is then parsed by a 1-predictive parsing algorithm.

The transformation of the input is done by a generalized sequential machine g which maps strings in T^* into strings in $\{[a, i] \mid a \in T \cup \{\$\}, R_i \in \pi\}^*$ by reading the input in reverse. Formally, let $x = a_1 \cdots a_s \in T^*$ be the input string to be processed; $g(a_s \cdots a_1) = [\$, i_{s+1}][a_s, i_s] \cdots [a_1, i_1]$ where $\epsilon \in Ri_{s+1}$ and $a_j \cdots a_s \in R_{ij}$ for $1 \leqslant j \leqslant s$. The construction of the gsm from the regular partition π is straightforward.

The resulting transformed input string can now be parsed using the 1-predictive parsing algorithm described below. The parser uses a transition function M whose construction depends on the sets L_{ij} of Section 3. These sets are used to distinguish the stack contents in order to choose the proper production to be used to expand the leftmost nonterminal.

We now construct the parser transition function M for the grammar $G = (N, T, P, S)$ and regular partition $\pi = \{R_1, \ldots, R_n\}$. Let m be the number of productions in P. Let $M_{ij} = (Q_{ij}, T, \delta_{ij}, q^0_{ij}, F_{ij})$ be the minimal deterministic finite automaton accepting L^R_{ij} for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$. Let $Q = \{[q_{1,1}, q_{1,2}, \ldots, q_{n,m}] \mid q_{i,j} \in Q_{ij}\}$ and let $q^0 = [q^0_{1,1}, q^0_{1,2}, \ldots, q^0_{n,m}]$. Define the function $\delta([q_{1,1}, \ldots, q_{n,m}], X) = [\delta_{1,1}(q_{1,1}, X), \ldots, \delta_{n,m}(q_{n,m}, X)]$ for $X \in N \cup T$ and the projections $h_{ij}([q_{1,1}, \ldots, q_{n,m}]) = q_{i,j}$ for $1 \leqslant i \leqslant n$ and $1 \leqslant j \leqslant m$. Let $\Gamma = \{[A, q] \mid A \in N, q \in Q\} \cup T \cup \{\$\}$ and $\Sigma = \{[a, i] \mid a \in T \cup \{\$\}, R_i \in \pi\}$. Define the parser transition function M: $\Gamma \times \Sigma \to (P \times \Gamma^* \cup \{accept, error, pop, undefined\})$ as follows:

$M([A, q], [a, i]) = (j, x_0[B_1, q_1]\, x_1[B_2, q_2] \cdots [B_s, q_s]x_s)$ if $x_k \in T^*$ for $0 \leqslant k \leqslant s$, $B_k \in N$ for $1 \leqslant k \leqslant s$ and there is one production j: $A \to x_0B_1x_1B_2 \cdots B_sx_s$ such that $h_{ij}(q) \in F_{ij}$ (where $q_k = \delta(q, x_sB_s \cdots B_{k+1}x_k)$ for $1 \leqslant k \leqslant s$).

$M([A, q], [a, i]) = error$ if there is no production j: $A \to \beta$ such that $h_{ij}(q) \in F_{ij}$.

$M([A, q], [a, i]) =$ ***undefined*** if there are two (or more) distinct productions j: $A \to \beta$ and k: $A \to \gamma$ such that $h_{ij}(q) \in F_{ij}$ and $h_{ik}(q) \in F_{ik}$.

$M(a, [b, i)] = pop$ if $a = b$.

$M(a, [b, i]) = error$ if $a \neq b$.

$M(a, [b, i)] = accept$ if $a = b = \$$.

DEFINITION 5.1. $[A, q]$ is accessible if there exists an α such that $S \overset{*}{\underset{\text{lm}}{\Rightarrow}} wA\alpha$ and $\delta(q^0, \alpha^R) = q$.

PROPOSITION 5.1. *G is not LL-regular if and only if M is undefined for some accessible $[A, q]$ and some $[a, i]$.*

Note that if $[A, q]$ is accessible then the symbols $[B_i, q_i]$ for $1 \leqslant i \leqslant s$ defined above are also accessible.

The LL-regular parsing algorithm is defined in terms of transitions from one configuration of the parser to another using the parser transition function M defined above. A configuration of the parser is either *error*, *accept* p or a three-tuple (x, α, p) where $x \in \Sigma^*$, $\alpha \in \Gamma^*$ and $p \in P^*$. The initial configuration of the parser is $(z', [S, q^0]\$, \epsilon)$ where $z' = g(z^R)^R$ (that is, z' is the result of gsm-mapping the original input string into one in Σ^*). The transition $\vdash$ is defined as:

$$(ax, X\alpha, p) \vdash (ax, \beta\alpha, pj) \quad \text{if} \quad M(X, a) = (j, \beta).$$
$$(ax, X\alpha, p) \vdash (x, \alpha, p) \quad \text{if} \quad M(a, X) = pop.$$
$$(ax, X\alpha, p) \vdash error \quad \text{if} \quad M(a, X) = error.$$
$$(ax, X\alpha, p) \vdash accept\ p \quad \text{if} \quad M(a, X) = accept.$$

LEMMA 5.2. *Let G be $LL(\pi)$. Let $h: \Gamma \to N \cup T$ be a homomorphism, where $h([A, q]) = A$, $h(a) = a$ and $h(\$) = \epsilon$. $(z', [S, q^0]\$, \epsilon) \vdash^* (x', [A, q]\beta', p)$ if and only if there is a derivation $S \underset{\text{lm}}{\overset{*}{\Rightarrow}} wA\beta \underset{\text{lm}}{\overset{*}{\Rightarrow}} wx = z$, where $\beta = h(\beta')$, $z' = g(z^R)^R$, $x' = g(x^R)^R$.*

THEOREM 5.3. *$(g(z^R)^R, [S, q^0]\$, \epsilon) \vdash^*$ accept p if and only if G is $LL(\pi)$ and $S \underset{\text{lm}}{\overset{p}{\Rightarrow}} z$ is a derivation in G.*

Two simple observations about the construction above can be used to produce a practical parser. The first observation is that $M([A, q], [a, i])$ is the same for all $a \in T$. The second observation is that a simple equality check is all that is required for $M(a, [a, i])$; $a = b$ implies *pop*, $a \neq b$ implies *error* and $a = b = \$$ implies *accept* p. These observations suggest the following construction of a parser:

A configuration is represented by an input head and a stack. The input head is positioned at the first symbol of the transformed input string, and the stack initially contains $[S, q^0]\$$ (where the leftmost symbol is considered to be at the top of the stack).

The function M is represented by a table with a row for each accessible symbol $[A, q]$ and a column for each R_i in π.

A transition is implemented as follows: if the top of the stack is \$ and the symbol under the input head is $[\$, i]$ for any i then stop and accept; if the top of the stack is the symbol a and the symbol under the input head is $[a, i]$ for any i then pop the stack and advance the input head; if the top of the stack is the symbol a and the symbol under the input head is $[b, i]$ for $a \neq b$ and any i then report error; if the top of the stack is $[A, q]$ and the symbol under the input head is $[a, i]$ for any a then consult row $[A, q]$ and column R_i of the table representing M: if *error* then report error, if (j, β) then replace $[A, q]$ on the stack by β and perform another transition.

6. SUMMARY

In this paper we have demonstrated that it is decidable whether or not a grammar is LL-regular for a specific regular partition, but that it is not decidable whether or not a regular partition exists for which a grammar is LL-regular. We have shown how to

construct a strong LL-regular grammar that generates the same language as a given LL-regular grammar. We have also presented an efficient, two-pass, top-down parsing algorithm for LL-regular grammars.

ACKNOWLEDGMENTS

The author is grateful to David Workman for his many helpful discussions and to Christoph Hoffmann for his ideas and suggestion during the preparation of this paper. The author also thanks the referees for many helpful suggestions.

REFERENCES

1. A. V. AHO AND J. D. ULLMAN, "The Theory of Parsing, Translation and Compiling," Vol. I Prentice–Hall, Englewood Cliffs, N.J., 1972.
2. K. CULIK, Contribution to deterministic top-down analysis of context-free languages, *Kybernetika (Prague)* (5) **4** (1968), 422–431.
3. K. CULIK AND R. COHEN, LR-regular grammars–An extension of LR(k) grammars, *J. Comput. System Sci.* **7** (1973), 66–96.
4. S. JARZABEK AND T. KRAWCZYK, LL-regular grammars, *Information Processing Lett.* (2) **4** (1975), 31–37.
5. D. E. KNUTH, Top down syntax analysis, *Acta Informatica* 1 (2) (1971), 79–110.
6. P. M. LEWIS, P. J. ROSENKRANTZ, AND R. E. STEARNS, Attributed translations, *J. Comput. System Sci.* **9** (1974), 279–307.
7. A. NIJHOLT, On the parsing of LL-regular grammars, *in* "Proceedings of the 5th Symposium on the Mathematical Foundations of Computer Science," pp. 446–452, 1976.
8. A. NIJHOLT, private communication.
9. W. OGDEN, private communication to K. Culik.